

1                   COMPILATION AND RUNTIME INFORMATION GENERATION  
2                                   AND OPTIMIZATION

3    **FIELD OF INVENTION**

4    The present invention relates to a compiler apparatus, program,  
5    record medium, and method, and to runtime information  
6    generating apparatus and program. More particularly, the  
7    present invention relates to the compiler apparatus, compiler  
8    program, record medium, compilation method, runtime information  
9    generating apparatus and runtime information generating program  
10   for performing optimization by using execution information  
11   obtained when a program is executed.

12   **BACKGROUND OF THE INVENTION**

13   In the past, a technology for collecting the number of times of  
14   execution of each of a plurality of execution paths of a  
15   program was used. For instance, according to the technology  
16   described in the Non-Patent Document 1, a compiler can insert a  
17   counter at an appropriate position in order to count the number  
18   of times of execution of the plurality of execution paths.

19   Non-Patent Document 1

20   "Efficient Path Profiling," Proceedings of 29th International  
21   Conference on Microarchitecture (MICRO-29), Ball, T and Larus,  
22   J. R., pp. 46 to 57, Dec. 1996

23   Problems to be solved by the invention

24   However, the above technology requires a long time to process a  
25   collection even though it can adequately collect the number of

1 times of execution of each execution path.

2 **SUMMARY OF THE INVENTION**

3 Thus, an object of the present invention is to provide the  
4 compiler apparatus, compiler program, record medium,  
5 compilation method, runtime information generating apparatus  
6 and runtime information generating program capable of solving  
7 the problem. This object is achieved by combining the  
8 characteristics described in independent sections of articles  
9 in the description. The subordinate sections prescribe  
10 further advantageous embodiments of the present invention.

11 To be more specific, a first form of the present invention  
12 provides a compiler apparatus, a compilation method, a compiler  
13 program, a runtime information generating apparatus, a runtime  
14 information generating program and a record medium for  
15 collecting frequencies with which each process is executed in a  
16 program to be optimized and optimizing the program based on the  
17 collected frequencies, having a loop process detection portion  
18 for detecting a repeatedly executed loop process of the  
19 program, a loop process frequency collection portion for  
20 collecting loop process frequencies with which the loop process  
21 is executed in the program, an in-loop process frequency  
22 collection portion for collecting in-loop process frequencies  
23 with which, as against the number of times of execution of the  
24 loop process, each of a plurality of in-loop processes included  
25 in the loop process is executed, an in-loop execution  
26 information generating portion for, based on the loop process  
27 frequencies and the in-loop process frequencies, generating  
28 in-loop execution information indicating the frequencies with  
29 which each of the plurality of in-loop processes is executed in  
30 the case where the program is executed, and an optimization

1 portion for optimizing the program based on the in-loop  
2 execution information generated by the in-loop execution  
3 information generating portion.

4 The above overview of the invention does not list all the  
5 necessary characteristics of the present invention, and  
6 sub-combinations of the characteristic group may also be  
7 inventions.

#### 8 **BRIEF DESCRIPTION OF THE DRAWINGS**

9 The invention and its embodiments will be more fully  
10 appreciated by reference to the following detailed description  
11 of advantageous and illustrative embodiments in accordance with  
12 the present invention when taken in conjunction with the  
13 accompanying drawings, in which:

14 Fig. 1 shows a functional block diagram of a compiler apparatus  
15 10;

16 Fig. 2 shows a flowchart of the compiler apparatus 10;

17 Fig. 3 shows an example of a program to be optimized;

18 Fig. 4 shows an example of a control flow graph;

19 Fig. 5 (a) shows an example of the control flow graph for which  
20 structure graphs will be generated;

21 Fig. 5 (b) shows execution paths of the control flow graph;

22 Fig. 5 (c) shows the execution paths of the structure graph  
23 generated from the control flow graph;

1 Fig. 6 (a) shows an example of an outline structure graph  
2 generated from the control flow graph shown in Fig. 4;

3 Fig. 6 (b) shows an example of an in-outer loop structure graph  
4 generated from the control flow graph shown in Fig. 4;

5 Fig. 6 (c) shows an example of an in-inner loop structure graph  
6 generated from the control flow graph shown in Fig. 4;

7 Fig. 7 (a) shows an example wherein a counter inserted into the  
8 program is stopped;

9 Fig. 7 (b) shows an example wherein the counter inserted into  
10 the program is started;

11 Fig. 7 (c) shows an example of generating a plurality of  
12 counters at the same insertion position;

13 Fig. 8 shows an example of execution information generated by  
14 the compiler apparatus 10;

15 Fig. 9 (a) shows the number of times of execution of each  
16 execution path determined by the outline structure graph;

17 Fig. 9 (b) shows the number of times of execution of each  
18 execution path determined by the in-outer loop structure graph;  
19 Fig. 9 (c) shows the number of times of execution of each  
20 execution path determined by the in-inner loop structure graph;

21 Fig. 9 (d) shows an example of in-loop execution information  
22 generated by an in-loop execution information generating  
23 portion 160;

1 Fig. 10 (a) shows an example wherein the program is optimized  
2 by an optimization portion 30;

3 Fig. 10 (b) shows the results wherein instruction sequences are  
4 placed in the program optimized by the optimization portion 30;

5 Fig. 11 shows an example of the execution information in a  
6 first other example;

7 Fig. 12 (a) shows an example of the execution information  
8 collected in the first other example on the control flow graph;

9 Fig. 12 (b) shows an example of the execution information  
10 collected in the first other example in a table;

11 Fig. 13 shows an example of the program optimized in a second  
12 other example; and

13 Fig. 14 shows an example of hardware configuration of the  
14 compiler apparatus 10 according to the embodiment described  
15 above.

16 **DESCRIPTION OF SYMBOLS**

17 10 ... Compiler apparatus  
18 20 ... Runtime information generating apparatus  
19 30 ... Optimization portion  
20 100 ... Control flow graph generating portion  
21 110 ... Loop detection portion  
22 120 ... Structure graph generating portion  
23 130 ... Counter insertion portion  
24 140 ... Loop process frequency collection portion  
25 150 ... In-loop process frequency collection portion

1        160 ... In-loop execution information generating portion  
2        500 ... Header node  
3        510 ... Latch node  
4        520 ... Execution path  
5        530 ... Execution path  
6        540 ... Execution path  
7        550 ... Execution path  
8        560 ... Execution path  
9        700 ... NOP instruction  
10       710 ... Determination process  
11       720 ... Jump instruction  
12       730 ... Determination process

13       **DETAILED DESCRIPTION OF THE INVENTION**

14       The present invention provides methods, systems and apparatus  
15       for compiler apparatus, compiler program, record medium,  
16       compilation method, runtime information generating apparatus  
17       and runtime information generating program capable of solving  
18       the problem of requiring a long time to process a collection  
19       even though it can adequately collect the number of times of  
20       execution of each execution path.

21       An example embodiment of the present invention provides a  
22       compiler apparatus, a compilation method, a compiler program, a  
23       runtime information generating apparatus, a runtime information  
24       generating program and a record medium for collecting  
25       frequencies with which each process is executed in a program to  
26       be optimized and optimizing the program based on the collected  
27       frequencies, having a loop process detection portion for  
28       detecting a repeatedly executed loop process of the program, a  
29       loop process frequency collection portion for collecting loop  
30       process frequencies with which the loop process is executed in

1 the program, an in-loop process frequency collection portion  
2 for collecting in-loop process frequencies with which, as  
3 against the number of times of execution of the loop process,  
4 each of a plurality of in-loop processes included in the loop  
5 process is executed, an in-loop execution information  
6 generating portion for, based on the loop process frequencies  
7 and the in-loop process frequencies, generating in-loop  
8 execution information indicating the frequencies with which  
9 each of the plurality of in-loop processes is executed in the  
10 case where the program is executed, and an optimization portion  
11 for optimizing the program based on the in-loop execution  
12 information generated by the in-loop execution information  
13 generating portion.

#### 14 Preferred embodiment

15 Hereafter, the present invention will be described through an  
16 embodiment. However, the following embodiment does not limit  
17 the invention according to the claims, and all the combinations  
18 described in the embodiment are not always essential to the  
19 means for solving the problem of the invention.

20 Figure 1 shows a functional block diagram of a compiler  
21 apparatus 10. The compiler apparatus 10 has a runtime  
22 information generating apparatus 20 for collecting frequencies  
23 with which each process is executed in a program to be  
24 optimized and an optimization portion 30 for optimizing the  
25 program based on the frequencies collected by the runtime  
26 information generating apparatus 20. The runtime information  
27 generating apparatus 20 has a control flow graph generating  
28 portion 100, a loop detection portion 110, a structure graph  
29 generating portion 120, a counter insertion portion 130, a loop  
30 process frequency collection portion 140, an in-loop process  
31 frequency collection portion 150 and an in-loop execution

1 information generating portion 160, and has the program  
2 optimized by the optimization portion 30 based on in-loop  
3 execution information generated by the in-loop execution  
4 information generating portion 160.

5 On receiving the program to be compiled, the control flow graph  
6 generating portion 100 generates each of a plurality of  
7 instruction sequences in the program as a node, and generates a  
8 control flow graph in which the execution order of the  
9 plurality of instruction sequences is generated as a directed  
10 edge of the nodes. And the control flow graph generating  
11 portion 100 sends the control flow graph to the loop detection  
12 portion 110 together with the program.

13 The program to be compiled is an intermediate expression  
14 generated from a source program for the sake of efficient  
15 optimization, which is a byte code of Java<sup>®</sup> for instance.  
16 Instead, the program may be either RTL (Registered Transfer  
17 Language) or a quadruplet expression.

18 The instruction sequence is a set of instructions to be  
19 consecutively executed. As an example, the instruction  
20 sequence is a basic block which is the set of instructions,  
21 wherein the instructions other than the instruction to be  
22 executed first and the instruction to be executed last are  
23 neither branching sources nor branching destinations of a  
24 branch instruction. As another example, the instruction  
25 sequence may be a super block which is the set of instructions,  
26 wherein the instructions other than the instruction to be  
27 executed first and the instruction to be executed last are not  
28 the branching destinations of the branch instruction

29 On receiving the control flow graph and the program from the  
30 control flow graph generating portion 100, the loop detection



1 portion 110 detects a repeatedly executed loop process of the  
2 program. In the case where the detected loop process includes  
3 an inner loop process which is a further inside loop process,  
4 the loop detection portion 110 further detects the inner loop  
5 process. And the loop detection portion 110 sends information  
6 on the detected loop process to the structure graph generating  
7 portion 120 together with the control flow graph and the  
8 program. The loop process is the set of instructions  
9 corresponding to strongly connected components which are a set  
10 of mutually reachable nodes in the control flow graph.

11 The structure graph generating portion 120 generates an outline  
12 structure graph in which an outer loop node is generated as a  
13 single node for showing an outer loop process in its entirety  
14 in the control flow graph instead of a collection of the nodes  
15 forming the outer loop process. The structure graph generating  
16 portion 120 also generates an in-outer loop structure graph in  
17 which an inner loop node is generated as a single node for  
18 showing an inner loop process in its entirety in the control  
19 flow graph of the outer loop process instead of a collection of  
20 the nodes forming the inner loop process. Furthermore, the  
21 structure graph generating portion 120 generates an in-inner  
22 loop structure graph which is the control flow graph of the  
23 inner loop process. And the structure graph generating portion  
24 120 sends the outline structure graph, in-outer loop structure  
25 graph, in-inner loop structure graph and program to the counter  
26 insertion portion 130.

27 The counter insertion portion 130 inserts the counter into the  
28 program in order to count the number of times of execution of  
29 each execution path in each of the outline structure graph,  
30 in-outer loop structure graph and in-inner loop structure  
31 graph. And the counter insertion portion 130 sends the program  
32 having the counter inserted therein to the loop process

1 frequency collection portion 140 together with the outline  
2 structure graph, in-outer loop structure graph and in-inner  
3 loop structure graph.

4 The loop process frequency collection portion 140 receives the  
5 outline structure graph, in-outer loop structure graph and  
6 in-inner loop structure graph from the counter insertion  
7 portion 130. In the case of receiving the program having the  
8 counter inserted therein from the counter insertion portion  
9 130, the loop process frequency collection portion 140 starts  
10 the inserted counter and executes the received program in order  
11 to count the number of times of execution of each execution  
12 path in the outline structure graph. Thereafter, the loop  
13 process frequency collection portion 140 stops the started  
14 counter when the program is executed a predetermined number of  
15 times. And the loop process frequency collection portion 140  
16 collects the number of times of execution of the outer loop  
17 process determined by the counter on stopping as outer loop  
18 process frequencies with which the outer loop process is  
19 executed, and sends the collection results to the in-loop  
20 process frequency collection portion 150 together with the  
21 program. The loop process frequency collection portion 140  
22 sends to the optimization portion 30, together with the  
23 program, outline structure graph frequency information  
24 indicating the frequency with which, as against the numbers of  
25 times of execution of the program, each execution path in the  
26 outline structure graph is executed.

27 Preferably, the loop process frequency collection portion 140  
28 detects a more frequently executed program piece by using an  
29 apparatus such as a timer sampling profiler for determining an  
30 execution frequency of the program, and starts the counter just  
31 for the outline structure graph of the program piece. Here,  
32 the program piece is a method, a function or a procedure for

1 instance. In this case, it is possible to optimize the more  
2 frequently executed program piece in preference so that  
3 processing speed of the program can be improved and the  
4 compiler apparatus 10 can be operated at high speed.

5 In the case of receiving the in-loop execution information on  
6 the outer loop process from the in-loop execution information  
7 generating portion 160, the loop process frequency collection  
8 portion 140 collects the loop process frequencies of the inner  
9 loop process based on the in-loop execution information and  
10 sends the collection results to the in-loop process frequency  
11 collection portion 150.

12 On receiving the collection results of the outer loop process  
13 frequencies from the loop process frequency collection portion  
14 140, the in-loop process frequency collection portion 150  
15 determines whether or not the process frequencies of the outer  
16 loop process are higher than a predetermined reference  
17 frequency. In the case where the process frequencies of the  
18 outer loop process are higher than the predetermined reference  
19 frequency, the in-loop process frequency collection portion 150  
20 starts the inserted counter in order to count the number of  
21 times of execution of each execution path in the in-outer loop  
22 structure graph, and thereby determines the number of times of  
23 execution of each of a plurality of in-loop processes in the  
24 outer loop process. Thereafter, the in-loop process frequency  
25 collection portion 150 stops the started counter when a total  
26 of determined values of the plurality of in-loop processes  
27 becomes a predetermined number of times. And the in-loop  
28 process frequency collection portion 150 collects as in-outer  
29 loop process frequencies the number of times of execution of  
30 each in-outer loop process as against the numbers of times for  
31 the in-outer loop processes to be executed based on the  
32 determined values of the stopped counter, and sends the

1 collection results to the in-loop execution information  
2 generating portion 160.

3 A description will be omitted as to the process in the case  
4 where the in-loop process frequency collection portion 150  
5 receives the collection results of the inner loop process  
6 frequencies from the loop process frequency collection portion  
7 140 because it is approximately the same as the process  
8 performed for the outer loop process by the in-loop process  
9 frequency collection portion 150.

10 Here, the in-loop process is the set of instructions on the  
11 execution path of the structure graph, for instance. Instead,  
12 the in-loop process may be either the instruction sequence  
13 indicated by each node of the structure graph or the branch  
14 instruction in the structure graph. To be more specific, the  
15 in-loop process frequency collection portion 150 generates a  
16 plurality of instruction groups from the instructions in the  
17 program by a predetermined method, and determines the number of  
18 times of execution of each instruction group as the number of  
19 times of execution of each in-loop process.

20 In the case of receiving the outer loop process frequencies and  
21 in-outer loop process frequencies, the in-loop execution  
22 information generating portion 160 generates the in-loop  
23 execution information for indicating the frequency with which  
24 each of the plurality of in-outer loop processes is executed in  
25 the case where the program is executed, and sends it to the  
26 loop process frequency collection portion 140 and optimization  
27 portion 30. In the case of receiving the inner loop process  
28 frequencies and in-inner loop process frequencies, the in-loop  
29 execution information generating portion 160 generates the  
30 in-loop execution information for indicating the frequency with  
31 which each of the plurality of inner loop processes is executed

1 in the case where the program is executed, and sends it to the  
2 optimization portion 30.

3 In the case where there is an inner loop process further inside  
4 the inner loop process, the in-loop execution information  
5 generating portion 160 may send the in-loop execution  
6 information on the inner loop process to the loop process  
7 process frequency collection portion 140. In this case, the loop  
8 process frequency collection portion 140 and the in-loop  
9 process frequency collection portion 150 repeat approximately  
10 the same operation as to the loop process further inside the  
11 inner loop process.

12 The optimization portion 30 optimizes the program received from  
13 the loop process frequency collection portion 140 based on the  
14 in-loop execution information and outline structure graph  
15 frequency information. Instead, the optimization portion 30  
16 may optimize the program before having the counter inserted by  
17 the counter insertion portion 130. And the optimization  
18 portion 30 outputs the optimized program as the program of the  
19 compilation results.

20 Figure 2 shows a flowchart of the compiler apparatus 10. On  
21 receiving the program to be compiled, the control flow graph  
22 generating portion 100 generates the control flow graph of the  
23 program (S200). And the loop detection portion 110 detects the  
24 repeatedly executed loop process of the program (S210). In the  
25 case where the detected loop process includes the inner loop  
26 process which is a further inside loop process, the loop  
27 detection portion 110 further detects the inner loop process.  
28 The structure graph generating portion 120 generates the  
29 outline structure graph in which the outer loop node is  
30 generated as the single node for showing the outer loop process  
31 in its entirety in the control flow graph instead of the

1 collection of the nodes forming the outer loop process (S220).  
2 The structure graph generating portion 120 also generates an  
3 in-outer loop structure graph in which the inner loop node is  
4 generated as the single node for showing the inner loop process  
5 in its entirety in the control flow graph of the outer loop  
6 process instead of the collection of the nodes forming the  
7 inner loop process. Furthermore, the structure graph  
8 generating portion 120 generates an in-inner loop structure  
9 graph which is the control flow graph of the inner loop  
10 process.

11 The counter insertion portion 130 inserts the counter into the  
12 program in order to count the number of times of execution of  
13 each execution path in each of the outline structure graph,  
14 in-outer loop structure graph and in-inner loop structure graph  
15 (S230). And the loop process frequency collection portion 140  
16 generates the process frequencies of the loop process, for  
17 example, the outer loop process frequencies for instance by  
18 executing the program (S240). The loop process frequency  
19 collection portion 140 generates the inner loop process  
20 frequencies based on the in-loop process frequencies of the  
21 outer loop process. In the case where the process frequencies  
22 of the loop process are higher than the predetermined  
23 frequencies (S250, YES), the in-loop process frequency  
24 collection portion 150 collects the in-loop process frequencies  
25 with which, as against the number of times of execution of the  
26 loop process, each of the plurality of in-loop processes in the  
27 loop process is executed (S260). The in-loop execution  
28 information generating portion 160 generates the in-loop  
29 execution information based on the loop process frequencies and  
30 the in-loop process frequencies (S270). In the case where the  
31 loop process includes the loop process further inside (S280,  
32 YES), the in-loop execution information generating portion 160  
33 shifts the process to S240 in order to collect the process

1 frequencies of the inner loop process.

2 In the case where the process frequencies of the loop process  
3 are lower than the predetermined frequencies (S250, NO), or in  
4 the case where the loop process includes no loop process  
5 further inside (S280, NO), the optimization portion 30  
6 optimizes the program based on the in-loop execution  
7 information, and outputs it as the program of the compilation  
8 results (S290).

9 The timing for the in-loop process frequency collection portion  
10 150 to collect the in-loop process frequencies is not limited  
11 to the timing in this flowchart. For instance, the in-loop  
12 process frequency collection portion 150 may start collecting  
13 the in-loop process frequencies as to each of the in-outer loop  
14 structure graph and in-inner loop structure graph when the loop  
15 process frequency collection portion 140 starts collecting the  
16 number of times of execution of each execution path in the  
17 outline structure graph. As another example, it is feasible to  
18 have the order of collecting the in-loop process frequencies  
19 predetermined between the in-outer loop structure graph and  
20 in-inner loop structure graph so that the in-loop process  
21 frequency collection portion 150 may collect the in-loop  
22 process frequencies in the predetermined order.

23 As a further example, in the case where a plurality of outer  
24 loop processes exist in the program, the in-loop process  
25 frequency collection portion 150 may start collecting the  
26 in-loop process frequencies for each depth of a hierarchy. For  
27 instance, in the case of starting collecting the in-loop  
28 process frequencies in one outer loop process, the in-loop  
29 process frequency collection portion 150 may start collecting  
30 the in-loop process frequencies in the other outer loop  
31 processes, and in the case of starting collecting the in-loop

1 process frequencies in one inner loop process, it may start  
2 collecting the in-loop process frequencies in the other inner  
3 loop processes.

4 Figure 3 shows an example of the program to be optimized. This  
5 program has a method "m" indicated by the sentences in the 1st  
6 to 14th lines. The method "m" has the outer loop process from  
7 the 4th to 12th lines. And the outer loop process has the  
8 inner loop process from the 6th to 10th lines. The sentence in  
9 the 9th line indicates the process for finishing the process of  
10 the method "m" in the case where the condition shown in the 8th  
11 line holds.

12 The sentence in the 3rd line, sentence in the 5th line,  
13 sentence in the 6th line, sentence in the 7th to 8th lines,  
14 sentence in the 9th line, sentence in the 11th to 12th lines,  
15 and sentence in the 13th line constitute the first to seven  
16 basic blocks respectively.

17 Figure 4 shows an example of the control flow graph. On  
18 receiving the program shown in Figure 3, the control flow graph  
19 generating portion 100 generates the control flow graph shown  
20 in Figure 4. In Figure 4, circles are the nodes indicating the  
21 instruction sequences of the program, and arrows are directed  
22 edges indicating the execution order of the instruction  
23 sequences. The instruction sequences in Figure 4 are the basic  
24 blocks, and the numbers described in the nodes are node numbers  
25 for identifying the basic blocks shown in Figure 3. By way of  
26 example, the directed edge from the third node to the fourth  
27 node and the sixth node indicates that the fourth or sixth  
28 basic block is executed after the third basic block.

29 Figure 5 (a) shows an example of the control flow graph for  
30 which the structure graphs will be generated. According to the



1 control flow graph shown in Figure 5 (a), a header node 500 and  
2 a latch node 510 are sequentially executed after a pre-loop  
3 process is performed. Subsequently, the header node 500 or  
4 post-loop process is performed according to processing results  
5 of the latch node 510. To be more specific, the header node  
6 500 and latch node 510 form the loop process.

7 Figure 5 (b) shows the execution paths of the control flow  
8 graph. The control flow graph shown in Figure 5 (a) has an  
9 execution path 520 for sequentially performing the header node  
10 500 and latch node 510 from the pre-loop process without  
11 repeatedly performing them and moving on to the post-loop  
12 process, an execution path 530 for sequentially performing the  
13 header node 500 and latch node 510 from the pre-loop process  
14 and moving on to the repeated processing, an execution path 540  
15 for having the latch node 510 further processed by the  
16 execution path from the latch node 510 to the header node 500,  
17 and an execution path 550 for sequentially performing the  
18 header node 500 and latch node 510 and then moving on to the  
19 post-loop process.

20 Figure 5 (c) shows the execution paths of the structure graph  
21 generated from the control flow graph. The structure graph  
22 generating portion 120 generates the outline structure graph  
23 and the in-loop structure graph. The in-loop structure graph  
24 has the execution path 530, execution path 540, execution path  
25 550 and a control flow 560 from the header node 500 to the  
26 latch node 510. The outline structure graph has the execution  
27 path 520 from the pre-loop process through the loop process to  
28 the post-loop process. Thus, the structure graph generating  
29 portion 120 generates as the outline structure graph the graph  
30 in which the loop process in its entirety is generated as the  
31 single loop node in the control flow graph instead of the  
32 collection of the nodes forming the loop process. To be more

1 specific, the execution path of the outline structure graph in  
2 Figure 5 (c) is the execution path 520. The structure graph  
3 generating portion 120 also generates the control flow graph of  
4 the collection of the nodes forming the loop process as the  
5 in-loop structure graph. To be more specific, the execution  
6 paths of the in-loop structure graph in Figure 5 (c) are the  
7 execution path 530, execution path 540, execution path 550 and  
8 a control flow 560.

9 To be more precise, the structure graph generating portion 120  
10 performs the following process in order to generate the in-loop  
11 structure graph. The structure graph generating portion 120  
12 generates as the in-loop structure graph the header node 500  
13 which is an entry node for starting the loop process from  
14 outside the loop process and the latch node 510 which is an  
15 exit node for moving the process from the loop process to  
16 outside the loop process. And in the control flow graph, the  
17 structure graph generating portion 120 includes all the edges  
18 and nodes from the header node 500 to the latch node 510 in the  
19 in-loop structure graph. And the structure graph generating  
20 portion 120 generates the edge on the header node 500 from a  
21 dummy node indicating a starting point of the in-loop structure  
22 graph instead of the pre-loop process. The structure graph  
23 generating portion 120 also generates the edge to the dummy  
24 node indicating an ending point of the in-loop structure graph  
25 from the latch node 510 instead of the post-loop process.

26 Figure 6 (a) shows an example of the outline structure graph  
27 generated from the control flow graph shown in Figure 4. The  
28 circles in a heavy line in Figure 6 (a) indicate the loop nodes  
29 generated instead of the loop process. For instance, the  
30 structure graph generating portion 120 generates the outline  
31 structure graph by the following process. In the control flow  
32 graph shown in Figure 4, the structure graph generating portion

1 120 generates a second node which is an outer loop node showing  
2 the entire outer loop process instead of a set of nodes forming  
3 the outer loop process, that is, the second, third, fourth and  
4 sixth nodes.

5 Subsequently, the structure graph generating portion 120  
6 generates the directed edge in a dotted line from the second  
7 node to the fifth and seventh nodes performed after the second  
8 node. Here, the directed edge in the dotted line does not  
9 really exist in the control flow graph shown in Figure 4, but  
10 it indicates a virtual execution route implemented by a  
11 combination of a plurality of directed edges in the control  
12 flow graph. For instance, the directed edge from the second  
13 node to the fifth node indicates the execution route leading to  
14 the fifth node via the fourth node after the execution of the  
15 outer loop. The virtual nodes indicating the starting point  
16 and ending point of the outline structure graph are indicated  
17 as E1 and X1 respectively.

18 Figure 6 (b) shows an example of the in-outer loop structure  
19 graph generated from the control flow graph shown in Figure 4.  
20 The structure graph generating portion 120 generates the  
21 in-outer loop structure graph by the following process. In the  
22 control flow graph of the outer loop process, the structure  
23 graph generating portion 120 generates the third node which is  
24 the inner loop node showing the entire inner loop process  
25 instead of a set of nodes constituting the inner loop process,  
26 that is, the third and fourth nodes.

27 And the structure graph generating portion 120 generates the  
28 directed edge in the dotted line from the starting point E2 of  
29 the in-outer loop structure graph to the second node which is  
30 an entrance to the outer loop process, the directed edge in the  
31 dotted line from the third node to the ending point X2 of the

1 in-outer loop structure graph, and the directed edge in the  
2 dotted line from the sixth node to X2.

3 Figure 6 (c) shows an example of the in-inner loop structure  
4 graph generated from the control flow graph shown in Figure 4.  
5 The structure graph generating portion 120 generates the  
6 in-inner loop structure graph by the following process. The  
7 structure graph generating portion 120 generates the control  
8 flow graph of the set of nodes constituting the inner loop  
9 process. And the structure graph generating portion 120  
10 generates the directed edge in the dotted line from the  
11 starting point E3 of the in-inner loop structure graph to the  
12 third node which is the entrance to the inner loop process, the  
13 directed edge in the dotted line from the third node to the  
14 ending point X3 of the in-inner loop structure graph, and the  
15 directed edge in the dotted line from the fourth node to X3.

16 In the case where, unlike the example in Figure 6 (c), the  
17 program to be compiled is an irreducible graph, the structure  
18 graph generating portion 120 generates the directed edge for  
19 each of a plurality of nodes which may be the starting point of  
20 the loop process from the node showing the starting point of  
21 each structure graph.

22 The counter insertion portion 130 inserts the counter into the  
23 program in order to count the number of times of execution of  
24 each execution path in each of the structure graphs generated  
25 as above. An example of a counter position inserted by the  
26 counter insertion portion 130 is indicated by a black point.  
27 For instance, in the outline structure graph, the counter  
28 insertion portion 130 inserts the counters into the directed  
29 edge from the fourth node to the fifth node and the directed  
30 edge from the sixth node to the seventh node. In the in-outer  
31 loop structure graph, the counter insertion portion 130 inserts

1 the counters into the directed edge from the first node to the  
2 second node, the directed edge from the fourth node to the  
3 fifth node, the directed edge from the sixth node to the second  
4 node and the directed edge from the sixth node to the seventh  
5 node. In the in-inner loop structure graph, the counter  
6 insertion portion 130 inserts the counters into the directed  
7 edge from the second node to the third node, the directed edge  
8 from the third node to the sixth node, and the directed edge  
9 from the fourth node to the fifth node.

10 The positions for inserting the counters are not limited to the  
11 examples in the drawing. For instance, the Non-Patent Document  
12 1 has a proposal of a method for efficiently determining the  
13 number of times of execution of each execution path, and so the  
14 positions for inserting the counters may be determined by using  
15 the method. To be more specific, the counter insertion portion  
16 130 inserts the counter at the position capable of counting the  
17 number of times of execution of each execution path in each  
18 structure graph. The counter insertion portion 130 may insert  
19 an initialization process for initializing the counter as  
20 required. In the case where a plurality of counters are  
21 inserted into the program, the counter insertion portion 130  
22 may further insert into the program the process for changing  
23 the counter to be determined of the plurality of counters. For  
24 instance, in the case where the counter insertion portion 130  
25 generates each of the plurality of counters as an array  
26 variable which is one counter, it may further insert into the  
27 program a process for changing a subscript of the array  
28 variable in order to change the counter to be determined. To  
29 be more specific, the counter insertion portion 130 inserts  
30 into the program the process for controlling the counters in  
31 order to count the number of times of execution of each  
32 execution path.

1 Figure 7 (a) shows an example wherein the counter inserted into  
2 the program is stopped. The counter insertion portion 130  
3 inserts an NOP instruction 700 at an insertion position of the  
4 program for inserting the counter in order to count the number  
5 of times of execution of each execution path of the structure  
6 graphs. And the counter insertion portion 130 generates a  
7 determination process 710 for determining the number of times  
8 of execution. The counter insertion portion 130 generates a  
9 jump instruction for moving the process to the instruction  
10 executed immediately after the NOP instruction 700 at a portion  
11 executed at the end of the determination process 710.

12 To describe it further in detail, the counter insertion portion  
13 130 inserts the NOP instruction 700 or a jump instruction 720  
14 into the basic block of the program to be compiled. However,  
15 there are the cases where the NOP instruction 700 or jump  
16 instruction 720 cannot be inserted into an existing basic block  
17 depending on the execution path to be determined. In such  
18 cases, the counter insertion portion 130 may generate a new  
19 basic block, that is, the basic block for inserting an  
20 instruction to implement the counter such as the NOP  
21 instruction 700 or jump instruction 720.

22 Figure 7 (b) shows an example wherein the counter inserted into  
23 the program is started. The in-loop execution information  
24 generating portion 160 generates the jump instruction 720 for  
25 causing the process to jump to the determination process 710  
26 instead of the NOP instruction 700. Thus, the in-loop  
27 execution information generating portion 160 can have the  
28 number of times of execution of the execution paths including  
29 the jump instruction 720 determined by the determination  
30 process 710.

31 Figure 7 (c) shows an example of generating a plurality of

1 counters at the same insertion position. A description will be  
2 given by using Figure 7 (c) as to the process of the compiler  
3 apparatus 10 in the case where the insertion position in the  
4 program for inserting the counter in order to determine the  
5 number of times of execution of each execution path of the  
6 outline structure graph is the same as the position in the  
7 program for inserting the counter in order to determine the  
8 number of times of execution of each execution path of the  
9 in-loop structure graph, and the counter of one, at the most,  
10 of the outline structure graph and in-loop structure graph is  
11 started.

12 The counter insertion portion 130 generates a plurality of  
13 determination processes for determining the number of times of  
14 execution of each execution path in each of the outline  
15 structure graph and the in-loop structure graph. For instance,  
16 the counter insertion portion 130 generates the determination  
17 process 710 for determining the number of times of execution of  
18 each execution path of the outline structure graph and a  
19 determination process 730 for determining the number of times  
20 of execution of each execution path of the in-loop structure  
21 graph. Furthermore, the counter insertion portion 130 inserts  
22 the jump instruction for jumping to the instruction executed  
23 following the insertion position of the counter (the position  
24 of the jump instruction 720 for instance) at the position  
25 executed at the end of each of the determination process 710  
26 and determination process 730.

27 The in-loop execution information generating portion 160  
28 generates the jump instruction 720 for causing the process to  
29 jump to the determination process 710 at the insertion position  
30 of the counter so as to have the number of times of execution  
31 of each execution path of the outline structure graph  
32 determined. The in-loop execution information generating

1 portion 160 also generates the jump instruction 720 for causing  
2 the process to jump to the determination process 730 at the  
3 insertion position of the counter so as to have the number of  
4 times of execution of each execution path of the in-loop  
5 structure graph determined. Thus, the counter insertion  
6 portion 130 sets the jump destination of the jump instruction  
7 at one of the plurality of determination processes so as to  
8 determine the number of times of execution of each execution  
9 path of both the outline structure graph and in-loop structure  
10 graph.

11 The compiler apparatus 10 operates approximately as shown in  
12 Figure 7 (c) as shown below, even in the case where the  
13 insertion position in the program for inserting the counter in  
14 order to determine the number of times of execution of each  
15 execution path of the in-outer loop structure graph is the same  
16 as the position in the program for inserting the counter in  
17 order to determine the number of times of execution of each  
18 execution path of the in-inner loop structure graph, and the  
19 counter of one, at the most, of the in-outer loop structure  
20 graph and in-inner loop structure graph is started.

21 To be more precise, the counter insertion portion 130 generates  
22 a plurality of determination processes for determining the  
23 number of times of execution of each execution path in each of  
24 the in-outer loop structure graph and in-inner loop structure  
25 graph. For instance, the counter insertion portion 130  
26 generates the determination process 710 for determining the  
27 number of times of execution of each execution path of the  
28 in-outer loop structure graph and a determination process 730  
29 for determining the number of times of execution of each  
30 execution path of the in-inner loop structure graph.  
31 Furthermore, the counter insertion portion 130 inserts the jump  
32 instruction for jumping to the instruction executed following



1 the insertion position of the counter (the position of the jump  
2 instruction 720 for instance) at the position executed at the  
3 end of each of the determination process 710 and determination  
4 process 730.

5 The in-loop execution information generating portion 160  
6 generates the jump instruction 720 for causing the process to  
7 jump to the determination process 710 at the insertion position  
8 of the counter so as to have the number of times of execution  
9 of each execution path of the in-outer loop structure graph  
10 determined. The in-loop execution information generating  
11 portion 160 also generates the jump instruction 720 for causing  
12 the process to jump to the determination process 730 at the  
13 insertion position of the counter so as to have the number of  
14 times of execution of each execution path of the in-inner loop  
15 structure graph determined. Thus, the counter insertion  
16 portion 130 sets the jump destination of the jump instruction  
17 at one of the plurality of determination processes so as to  
18 determine the number of times of execution of each execution  
19 path of both the in-outer loop structure graph and in-inner  
20 loop structure graph.

21 As described above, the counter insertion portion 130 can  
22 determine the number of times of execution of the execution  
23 paths of both the structure graphs at the insertion positions  
24 as shown in the drawing in the case where the counter of one,  
25 at the most, of the two structure graphs is started, that is,  
26 in the case where it is assured that the counters are not  
27 simultaneously used in both the structure graphs. In the case  
28 of three or more structure graphs, the compiler apparatus 10  
29 can share the counter likewise when the counter of one  
30 structure graph is started at the most.

31 As for the two counters simultaneously used, the counter

1 insertion portion 130 generates each of the two counters at the  
2 insertion position. For instance, in the case of starting to  
3 collect the in-loop process frequencies in the in-outer loop  
4 structure graph when the collection is started as to the number  
5 of times of execution of each execution path of the outline  
6 structure graph, the counter insertion portion 130 generates  
7 each of the counters of each of the outline structure graph and  
8 in-outer loop structure graph at the insertion position. Thus,  
9 the counter insertion portion 130 may change the method of  
10 inserting the counters according to the timing for collecting  
11 the in-loop process frequencies.

12 In the case where no exclusive control is exerted other than  
13 the determination process and a plurality of threads  
14 simultaneously perform the determination process, the value of  
15 the counter may become incorrect. However, in the case where  
16 the number of threads is sufficiently smaller than the  
17 determined value, an error in the determined value is so slight  
18 that the compiler apparatus 10 can almost exactly determine the  
19 number of times of execution of each execution path.

20 Figure 8 shows an example of the execution information  
21 generated by the compiler apparatus 10. To describe it further  
22 in detail, Figure 8 associates an identification number for  
23 identifying the execution path with a permutation of the nodes  
24 constituting the execution path, the determined value which is  
25 the number of times of execution of the execution path  
26 determined by the counter, the execution information (in-loop  
27 execution information, for instance) generated based on the  
28 determined value, and the number of times of actual execution  
29 so as to show it in each structure graph.

30 The loop process frequency collection portion 140 stops the  
31 counter for determining the number of times of execution of the

1 execution path of the outline structure graph when having  
2 executed the program 100 times as predetermined. At this time,  
3 it sequentially executes from a node E1 to the first node,  
4 second node and seventh node, and the number of times of  
5 execution of the first execution path leading to the node X1 is  
6 determined as 100 times. To be more specific, the second  
7 execution path is not executed at all. In this case, the  
8 in-loop execution information generating portion 160 generates  
9 100.0 as the execution information which is the frequency with  
10 which the first path is executed in the case where the program  
11 is executed 100 times.

12 Subsequently, in the case where the outer loop process  
13 frequency, that is, the frequency with which the second node is  
14 executed is higher than the predetermined frequency, the  
15 in-loop process frequency collection portion 150 collects the  
16 in-outer loop process frequencies. First, the in-loop process  
17 frequency collection portion 150 starts the counter for  
18 determining the number of times of execution of each of the  
19 plurality of execution paths in the in-outer loop structure  
20 graph, and stops it when the total of determined values of the  
21 plurality of execution paths becomes 100 times as  
22 predetermined. And the in-loop process frequency collection  
23 portion 150 collects the in-outer loop process frequencies  
24 which is the frequency with which each execution path is  
25 executed as against the number of times of execution of the  
26 outer loop process.

27 For instance, the number of times of execution of the outer  
28 loop process is the number of times of moving the process from  
29 an E2 node to the outer loop, and so it is 51 times as the  
30 total value from the third path to the fifth path. The number  
31 of times of execution of the eighth path is 48 times, for  
32 instance. To be more specific, the in-loop process frequency

1 collection portion 150 collects the information indicating that  
2 the eighth execution path is executed 48 times in the case  
3 where the outer loop process is executed 51 times as the  
4 in-outer loop process frequency.

5 And the in-loop execution information generating portion 160  
6 generates 94.1 which is the in-loop execution information on  
7 the outer loop process by multiplying 100.0 as the process  
8 frequency of the outer loop process by the in-outer loop  
9 process frequency, for instance,  $48/51$  as the process frequency  
10 of the eighth execution path for instance. The in-loop  
11 execution information generating portion 160 also generates the  
12 in-loop execution information from the third path to the  
13 seventh path by approximately the same method as with the  
14 eighth execution path, and so a description thereof will be  
15 omitted.

16 Subsequently, the loop process frequency collection portion 140  
17 calculates the frequency with which the inner loop process  
18 frequency, that is, the third node is executed based on the  
19 in-loop execution information on the outer loop process. For  
20 instance, the loop process frequency collection portion 140  
21 selects all the execution paths for executing the third node in  
22 the in-outer loop structure graph, that is, the third to eighth  
23 paths. And the loop process frequency collection portion 140  
24 generates 196.1 which is the total value of the in-loop  
25 execution information in the selected paths as the inner loop  
26 process frequency.

27 Subsequently, in the case where the inner loop process  
28 frequency, that is, the frequency with which the third node is  
29 executed is higher than the predetermined frequency, the  
30 in-loop process frequency collection portion 150 collects the  
31 in-inner loop process frequencies by the following process.

1 The in-loop process frequency collection portion 150 starts the  
2 counter for determining the number of times of execution of  
3 each of the plurality of execution paths in the in-inner loop  
4 structure graph, and stops it when the total of determined  
5 values of the plurality of execution paths becomes 100 times as  
6 predetermined. And the in-loop process frequency collection  
7 portion 150 collects the in-inner loop process frequencies  
8 which is the frequency with which each execution path is  
9 executed as against the number of times of execution of the  
10 inner loop process.

11 For instance, the number of times of execution of the inner  
12 loop process is the number of times of moving the process from  
13 an E3 node to the inner loop, and so it is 58 times as the  
14 total value from the ninth path to the eleventh path. The  
15 number of times of execution of the thirteenth execution path  
16 is 40 times, for instance. To be more specific, the in-loop  
17 process frequency collection portion 150 collects the  
18 information indicating that the thirteenth execution path is  
19 executed 40 times in the case where the inner loop process is  
20 executed 58 times as the in-inner loop process frequency.

21 And the in-loop execution information generating portion 160  
22 generates 135.2 which is the in-loop execution information on  
23 the inner loop process by multiplying 196.1 as the process  
24 frequency of the inner loop process by the in-inner loop  
25 process frequency, for instance,  $40/58$  as the process frequency  
26 of the thirteenth execution path for instance. The in-loop  
27 execution information generating portion 160 also generates the  
28 in-loop execution information from the ninth path to the  
29 twelfth path and fourteenth path by approximately the same  
30 method as with the thirteenth execution path, and so a  
31 description thereof will be omitted.

1 The method of generating the in-loop execution information  
 2 described above will be indicated by a formula.

3 The in-loop execution information generating portion 160  
 4 generates the execution information on each execution path in a  
 5 structure graph X by multiplying the determined value which is  
 6 the number of times of execution of each execution path by a  
 7 correction coefficient Cx shown by the following formula.

8 [Formula 1]

$$9 \quad C_x = \begin{cases} \text{thresholdCount}(X) \cdots \cdots \cdots \text{In the case where } X \text{ is the outline} \\ \text{structure graph} \\ \frac{C_r \sum_{q \in P_Y(N_x)} C_q}{\sum_{q \in P_X(\text{entry})} C_p} \cdots \cdots \cdots \text{Otherwise} \end{cases}$$

10 Here, thresholdCount (X) is preset by associating it with the  
 11 structure graph X, and shows the total value of the determined  
 12 values collected in the structure graph X. Cp represents the  
 13 determined value of the number of times of execution collected  
 14 for a route p, Px (a) represents a collection of the routes  
 15 running through a node a in the structure graph X, Px (entry)  
 16 represents a collection of the routes entering the loop from  
 17 outside it in the structure graph X, and Nx is a loop node in  
 18 the structure graph of a high order hierarchy corresponding to  
 19 the structure graph X respectively. Cy is the correction  
 20 coefficient in the structure graph of the high order hierarchy  
 21 of the structure graph X. Here, the high order hierarchy is  
 22 the structure graph in a further outer loop process, for  
 23 instance. For instance, the high order hierarchy of the  
 24 in-inner loop structure graph is the in-outer loop structure  
 25 graph, and the high order hierarchy of the in-outer loop  
 26 structure graph is the outline structure graph.

27 This drawing further shows the number of times of actual  
 28 execution of each execution path determined by another method

1 by associating it to the execution information. The other  
2 method determines the number of times of execution of each  
3 execution path in the case of executing the program 10,000  
4 times.

5 As opposed to this, the compiler apparatus 10 according to this  
6 embodiment can generate approximately the same execution  
7 information as the number of times of actual execution by  
8 determining the number of times of execution of the execution  
9 paths 100 times for each structure graph, that is, 300 times in  
10 total. Accordingly, the compiler apparatus 10 can reduce the  
11 time required for the compilation process.

12 Figure 9 (a) shows the number of times of execution of each  
13 execution path determined by the outline structure graph. In  
14 the case where the program is executed 100 times, the first  
15 execution path leading to the node X1 from the node E1 by way  
16 of the first node, second node and seventh node is executed 100  
17 times.

18 Figure 9 (b) shows the number of times of execution of each  
19 execution path determined by the in-outer loop structure graph.  
20 In the case where the total of the number of times of execution  
21 of each execution path becomes 100 times, the fourth execution  
22 path leading to a node X2 from the node E2 by way of the second  
23 node, third node and sixth node is executed 50 times. The  
24 eighth execution path leading to the sixth node by way of the  
25 sixth node, second node and third node is executed 48 times.

26 Figure 9 (c) shows the number of times of execution of each  
27 execution path determined by the in-inner loop structure graph.  
28 In the case where the total of the number of times of execution  
29 of each execution path becomes 100 times, the eleventh  
30 execution path leading to a node X3 from the node E3 by way of

1 the third node is executed 56 times. The thirteen execution  
2 path leading to the fourth node by way of the fourth node and  
3 third node is executed 40 times.

4 Figure 9 (d) shows an example of the in-loop execution  
5 information generated by the in-loop execution information  
6 generating portion 160. The in-loop execution information  
7 generating portion 160 generates 94.1 as the in-loop execution  
8 information indicating the frequency with which the eighth  
9 execution path is executed in the case where the program is  
10 executed 100 times. The in-loop execution information  
11 generating portion 160 also generates 98.0 as the in-loop  
12 execution information indicating the frequency with which the  
13 fourth execution path is executed in the case where the program  
14 is executed 100 times. The in-loop execution information  
15 generating portion 160 also generates 135.2 as the in-loop  
16 execution information indicating the frequency with which the  
17 thirteenth execution path is executed in the case where the  
18 program is executed 100 times. To be more specific, according  
19 to the compiler apparatus 10, the program to be compiled  
20 executes the fourth execution path for continuously executing  
21 the program from the starting point to the ending point without  
22 performing the loop process, the eighth execution path for  
23 repeating the outer loop process, and the thirteenth execution  
24 path for repeating the inner loop process more frequently than  
25 other execution paths.

26 Figure 10 (a) shows an example wherein the program is optimized  
27 by the optimization portion 30. The optimization portion 30  
28 optimizes each of the plurality of execution paths more  
29 frequently executed (hot paths) based on the in-loop execution  
30 information to place them in contiguous areas. For instance,  
31 the optimization portion 30 separates the outer loop process  
32 and inner loop process in order to efficiently optimize the



1 fourth execution path for consecutively executing the first  
2 node, second node, third node, sixth node and seventh node.  
3 The optimization portion 30 separates the second node, third  
4 node, sixth node and seventh node shaded respectively from the  
5 control flow graph as the outer loop process. The optimization  
6 portion 30 performs loop peeling to the outer loop process so  
7 as to separate the third node and sixth node shaded  
8 respectively from the control flow graph as the inner loop  
9 process.

10 Figure 10 (b) shows the results wherein the instruction  
11 sequences are placed in the program optimized by the  
12 optimization portion 30. The optimization portion 30 places  
13 the instruction sequences from the first node to the seventh  
14 node, from the second node to the seventh node, and from the  
15 fourth node to the sixth node in the contiguous areas  
16 respectively. Branching processes performed not to be  
17 contiguously placed are shown by arrows. The execution  
18 information in each branching process is added to the arrow.  
19 As shown in Figure 10 (b), the optimization portion 30 can  
20 reduce the frequency with which the branching process is  
21 performed by the branch instruction. Thus, it is possible to  
22 improve efficiency of a branching forecast process by hardware.  
23 Furthermore, it improves a percent hit rate of a cache memory  
24 for instructions in a processor. It is also possible to  
25 decrease the number of redundant unconditional branches and  
26 redundant forward branches.

27 Figure 11 shows the method of generating the execution  
28 information in a first other example. The compiler apparatus  
29 in this example does not create the structure graph but  
30 determines the execution frequency as to all the execution  
31 paths of the control flow graph. For instance, the compiler  
32 apparatus in this example inserts the counters at the positions

1 of the black points in Figure 11, that is, into each of the  
2 edge from the first node to the second node, the edge from the  
3 fourth node to the third node, the edge from the fourth node to  
4 the fifth node, the edge from the sixth node to the second  
5 node, and the edge from the sixth node to the seventh node so  
6 as to collect the determined value on each counter.

7 Figure 12 (a) shows an example of the execution information  
8 collected in the first other example on the control flow graph.  
9 Figure 12 (b) shows an example of the execution information  
10 collected in the first other example in a table. The compiler  
11 apparatus in this example stops the counter when having  
12 executed the program 300 times in order to reduce the time  
13 required for the compilation. As shown in the drawing, the  
14 compiler apparatus in this example can detect that the third  
15 execution path leading to the seventh node from the first node  
16 by way of the second node, third node and sixth node, and the  
17 sixth execution path leading to the third node from the fourth  
18 node by way of the third node and fourth node are the hot paths  
19 more frequently executed than other execution paths. However,  
20 the number of times of execution of the program is small, and  
21 so the compiler apparatus in this example cannot detect that  
22 the twelfth execution path leading to the second node from the  
23 sixth node by way of the second node, third node and sixth node  
24 is the hot path. Therefore, unlike the example in Figure 10  
25 (b), it cannot perform the optimization for consecutively  
26 placing the second node, third node, sixth node and seventh  
27 node.

28 As opposed to this, the compiler apparatus 10 according to this  
29 embodiment can detect that the twelfth execution path is the  
30 hot path while reducing the time required for the compilation  
31 as with the first other example.

1 Figure 13 shows an example of the program optimized in a second  
2 other example. The compiler apparatus in this example collects  
3 the execution information of the program by an edge profile  
4 method of determining the number of times of processing each  
5 directed edge in the control flow graph. In this example, the  
6 execution path leading to the seventh node from the first node  
7 by way of the second node, third node and sixth node and the  
8 execution path leading to the fifth node from the fourth node  
9 are hot paths, and they are placed as consecutive instruction  
10 sequences respectively. However, it is not efficient because  
11 the forward branch instruction leading to the fourth node from  
12 the third node and the branch instruction leading to the third  
13 node from the fourth node are generated.

14 As opposed to this, according to Figure 10 (b), the program  
15 optimized by the compiler apparatus 10 has no forward branch  
16 instruction which is frequently executed, and so its execution  
17 efficiency is high.

18 Figure 14 shows an example of hardware configuration of the  
19 compiler apparatus 10 according to the embodiment described  
20 above. The compiler apparatus 10 related to the embodiment or  
21 a deformation example is equipped with a CPU peripheral portion  
22 having a CPU 1000, an RAM 1020, a graphic controller 1075 and a  
23 display device 1080 mutually connected by a host controller  
24 1082, an input-output portion having a communication interface  
25 1030, a hard disk drive 1040 and a CD ROM drive 1060 connected  
26 to the host controller 1082 by an input-output controller 1084,  
27 and a legacy input-output portion having an ROM 1010, a  
28 flexible disk drive 1050 and an input-output chip 1070  
29 connected to the input-output controller 1084.

30 The host controller 1082 connects the RAM 1020 to the CPU 1000  
31 and graphic controller 1075 accessing the RAM 1020 at a high

1 transfer rate. The CPU 1000 operates based on a compiler  
2 program and a runtime information generating program stored in  
3 the ROM 1010 and RAM 1020 so as to control each portion. The  
4 graphic controller 1075 obtains image data generated on a frame  
5 buffer provided in the RAM 1020 by the CPU 1000 and so on, and  
6 displays it on the display device 1080. Instead, the graphic  
7 controller 1075 may include therein the frame buffer for  
8 storing the image data generated by the CPU 1000 and so on.

9 The input-output controller 1084 connects the host controller  
10 1082 to the communication interface 1030, hard disk drive 1040  
11 and CD ROM drive 1060 which are relatively high-speed  
12 input-output devices. The communication interface 1030  
13 communicates with other apparatuses via a network. The hard  
14 disk drive 1040 stores the compiler program or runtime  
15 information generating program and the data used by the  
16 compiler apparatus 10. The CD ROM drive 1060 reads the  
17 compiler program, runtime information generating program or the  
18 data from a CD-ROM 1095, and submits it to the input-output  
19 chip 1070 via the RAM 1020.

20 The input-output controller 1084 has the ROM 1010 and  
21 relatively low-speed input-output devices such as the flexible  
22 disk drive 1050 and input-output chip 1070 connected thereto.  
23 The ROM 1010 stores a boot program executed by the CPU 1000 on  
24 starting the compiler apparatus 10, the program dependent on  
25 the hardware of the compiler apparatus 10 and so on. The  
26 flexible disk drive 1050 reads the compiler program or runtime  
27 information generating program or the data from a flexible disk  
28 1090, and provides it to the input-output chip 1070 via the RAM  
29 1020. The input-output chip 1070 connects various input-output  
30 devices via the flexible disk 1090 and a parallel port, a  
31 serial port, a keyboard port, a mouse port and so on, for  
32 instance.

1 The compiler program or runtime information generating program  
2 provided to the compiler apparatus 10 is stored in a record  
3 medium such as the flexible disk 1090, CD-ROM 1095 or an IC  
4 card, and is provided to a user. The compiler program or  
5 runtime information generating program is read from the record  
6 medium, and is installed on the compiler apparatus 10 via the  
7 input-output chip 1070 so as to be executed on the compiler  
8 apparatus 10.

9 The compiler program or runtime information generating program  
10 to be installed and executed on the compiler apparatus 10  
11 includes a control flow graph generation module, a loop  
12 detection module, a structure graph generation module, a  
13 counter insertion module, a loop process frequency collection  
14 module, an in-loop process frequency collection module, an  
15 in-loop execution information generating module and an  
16 optimization module. The operations performed by the compiler  
17 apparatus 10 being prompted by the modules are the same as the  
18 operations of corresponding members of the compiler apparatus  
19 10 described by referring to Figures 1 to 13, and so a  
20 description thereof will be omitted.

21 The program or modules described above may be stored on an  
22 external storage medium. As for the storage medium, in  
23 addition to the flexible disk 1090 and CD-ROM 1095, an optical  
24 record medium such as a DVD or a PD, a magneto-optical record  
25 medium such as an MD, a tape medium or a semiconductor memory  
26 such as the IC card may be used. It is also feasible to use as  
27 the record medium a storage device such as a hard disk or an  
28 RAM provided on a server system connected to a dedicated  
29 communication network or the Internet so as to provide the  
30 compiler program or runtime information generating program to  
31 the compiler apparatus 10 via the network.

1 As is clear from the above description, the compiler apparatus  
2 10 can collect the in-loop execution information at high speed  
3 and appropriately optimize the program. For instance, in the  
4 case where the compiler apparatus 10 is a runtime compiler, the  
5 program can be more efficiently optimized because the  
6 compilation cannot take so much time.

7 Although the present invention was described by using the  
8 embodiment above, the technical scope of the present invention  
9 is not limited to the scope of the above embodiment. It is  
10 possible to add various modifications and improvements to the  
11 above embodiment. It is clear from the description in claims  
12 that the embodiments having such modifications and improvements  
13 added thereto are included in the technical scope of the  
14 present invention.

15 According to the embodiment described above, the compiler  
16 apparatus, compiler program, record medium, compilation method,  
17 runtime information generating apparatus and runtime  
18 information generating program described in the articles are  
19 implemented.

20 (Article 1) A compiler apparatus for collecting the frequencies  
21 with which each process is executed in the program to be  
22 optimized and optimizing the above described program based on  
23 the collected frequencies, the above described apparatus having  
24 a loop process detection portion for detecting a repeatedly  
25 executed loop process of the above described program, a loop  
26 process frequency collection portion for collecting loop  
27 process frequencies with which the above described loop process  
28 is executed in the above described program, an in-loop process  
29 frequency collection portion for collecting in-loop process  
30 frequencies with which, as against the number of times of

1 execution of the above described loop process, each of a  
2 plurality of in-loop processes included in the above described  
3 loop process is executed, an in-loop execution information  
4 generating portion for, based on the above described loop  
5 process frequencies and the above described in-loop process  
6 frequencies, generating in-loop execution information  
7 indicating the frequencies with which each of the above  
8 described plurality of in-loop processes is executed in the  
9 case where the above described program is executed, and

10 an optimization portion for optimizing the above described  
11 program based on the above described in-loop execution  
12 information generated by the above described in-loop execution  
13 information generating portion.

14 (Article 2) The compiler apparatus according to article 1,  
15 wherein the above described in-loop process frequency  
16 collection portion collects the above described in-loop process  
17 frequencies in the case where the above described loop process  
18 frequencies are higher than a predetermined frequency.

19 (Article 3) The compiler apparatus according to article 1,  
20 wherein the above described in-loop execution information  
21 generating portion generates the above described in-loop  
22 execution information by multiplying the above described loop  
23 process frequencies by the above described in-loop process  
24 frequencies.

25 (Article 4) The compiler apparatus according to article 1,  
26 wherein the above described loop process is the outer loop  
27 process including the inner loop process which is a further  
28 inside loop process, the above described loop process detection  
29 portion further detects the above described inner loop process,  
30 the above described loop process frequency collection portion

1 further collects the loop process frequencies with which the  
2 above described inner loop process is executed in the above  
3 described program based on the above described in-loop  
4 execution information, the above described in-loop process  
5 frequency collection portion collects the in-loop process  
6 frequencies of the above described inner loop process, and the  
7 above described in-loop execution information generating  
8 portion generates the in-loop execution information on the  
9 above described inner loop process by multiplying the in-loop  
10 process frequencies in the above described inner loop process  
11 by the above described loop process frequencies of the above  
12 described inner loop process.

13 (Article 5) The compiler apparatus according to article 1,  
14 wherein the above described loop process frequency collection  
15 portion stops the counter for determining the number of times  
16 of execution of the above described loop process when the above  
17 described program is executed a predetermined number of times  
18 so as to collect the number of times determined by the counter  
19 as the above described loop process frequencies, and the above  
20 described in-loop process frequency collection portion stops  
21 the counter for determining the number of times of execution of  
22 each of the above described plurality of in-loop processes when  
23 a total of determined values of the above described plurality  
24 of in-loop processes becomes the predetermined number of times.

25 (Article 6) The compiler apparatus according to article 1,  
26 further having the control flow graph generating portion for  
27 generating the control flow graph in which each of a plurality  
28 of instruction sequences in the above described program is  
29 generated as a node and an execution order of the above  
30 described plurality of instruction sequences is generated as  
31 the directed edge of the above described nodes, a structure  
32 graph generating portion for, in the above described control



1 flow graph, generating an outline structure graph in which a  
2 single loop node for showing the above described loop process  
3 in its entirety is generated instead of the collection of the  
4 nodes forming the above described loop process and the in-loop  
5 structure graph which is the control flow graph of the  
6 collection of the nodes forming the above described loop  
7 process, and a counter insertion portion for, in each of the  
8 above described outline structure graph and the above described  
9 in-loop structure graph, inserting the counter into the above  
10 described program in order to count the number of times of  
11 execution of each execution path in the structure graphs, and  
12 wherein the above described loop process frequency collection  
13 portion generates as the above described loop process  
14 frequencies the numbers of times of execution of the above  
15 described loop node as against the numbers of times of  
16 execution of the above described program, and the above  
17 described in-loop process frequency collection portion collects  
18 as the above described in-loop process frequencies the number  
19 of times of execution of each execution path in the above  
20 described in-loop structure graph as against the numbers of  
21 times of execution of the above described loop process.

22 (Article 7) The compiler apparatus according to article 6,  
23 wherein in the case where the above described program is  
24 executed a predetermined number of times, the above described  
25 loop process frequency collection portion collects as the loop  
26 process frequencies the determined values of the counter  
27 inserted for counting the number of times of execution of the  
28 execution paths including the above described loop node, and in  
29 the case where a total of the determined values of the above  
30 described plurality of in-loop processes becomes a  
31 predetermined number of times, the above described in-loop  
32 process frequency collection portion collects the in-loop  
33 process frequencies based on the determined values of the

1 counter inserted for counting the number of times of execution  
2 of each execution path in the above described in-loop structure  
3 graph.

4 (Article 8) The compiler apparatus according to article 6,  
5 wherein, in the case where the insertion position in the above  
6 described program for inserting the counter for determining the  
7 number of times of execution of each execution path in the  
8 above described outline structure graph is the same as the  
9 position in the above described program for inserting the  
10 counter for determining the number of times of execution of  
11 each execution path in the above described in-loop structure  
12 graph and then the counter of one, at the most, of the above  
13 described outline structure graph and the above described  
14 in-loop structure graph is started, the above described counter  
15 insertion portion inserts into the insertion position the  
16 counter for determining the numbers of times of execution of  
17 the execution paths in both the above described outline  
18 structure graph and the above described in-loop structure  
19 graph.

20 (Article 9) The compiler apparatus according to article 6,  
21 wherein, in the case where the insertion position in the above  
22 described program for inserting the counter for determining the  
23 number of times of execution of each execution path in the  
24 above described outline structure graph is the same as the  
25 position in the above described program for inserting the  
26 counter for determining the number of times of execution of  
27 each execution path in the above described in-loop structure  
28 graph and then the counter of one, at the most, of the above  
29 described outline structure graph and the above described  
30 in-loop structure graph is started, the above described counter  
31 insertion portion generates a plurality of determination  
32 processes for determining the number of times of execution of

1 each execution path in each of the above described outline  
2 structure graph and the above described in-loop structure  
3 graph, and the above described in-loop process frequency  
4 collection portion inserts a jump instruction for moving the  
5 process to another portion into the above described insertion  
6 position and sets the jump destination of the jump instruction  
7 at one of the above described plurality of determination  
8 processes so as to determine the numbers of times of execution  
9 of the execution paths in both the above described outline  
10 structure graph and the above described in-loop structure  
11 graph.

12 (Article 10) The compiler apparatus according to article 6,  
13 wherein the above described loop process is the outer loop  
14 process including an inner loop process which is a further  
15 inside loop process, the above described loop process detection  
16 portion further detects the above described inner loop process,  
17 in the control flow graph of the above described outer loop  
18 process, the above described structure graph generating portion  
19 generates as an in-outer loop structure graph a graph in which  
20 the single inner loop node is generated instead of the  
21 collection of the nodes forming the above described inner loop  
22 process and generates the in-inner loop structure graph which  
23 is the control flow graph of the collection of the nodes  
24 forming the above described inner loop process, and the above  
25 described counter insertion portion further inserts the counter  
26 for determining the number of times of execution of each  
27 execution path in the above described in-inner loop structure  
28 graph, the above described loop process frequency collection  
29 portion further collects the loop process frequencies with  
30 which the above described inner loop process is executed in the  
31 above described program based on the above described in-loop  
32 execution information, the above described in-loop process  
33 frequency collection portion collects the frequencies of

1 execution of each execution path in the above described  
2 in-inner loop structure graph as the in-loop process  
3 frequencies of the above described inner loop process as  
4 against the number of times of execution of the above described  
5 inner loop process, and the above described in-loop execution  
6 information generating portion further generates the in-loop  
7 execution information on the above described inner loop process  
8 by multiplying the in-loop process frequencies in the above  
9 described inner loop process by the loop process frequencies of  
10 the above described inner loop process.

11 (Article 11) The compiler apparatus according to article 10,  
12 wherein, in the case where the insertion position in the above  
13 described program for inserting the counter for determining the  
14 number of times of execution of each execution path in the  
15 above described in-outer loop structure graph is the same as  
16 the position in the above described program for inserting the  
17 counter for determining the number of times of execution of  
18 each execution path in the above described in-inner loop  
19 structure graph and then the counter of one, at the most, of  
20 the above described in-outer loop structure graph and the above  
21 described in-inner loop structure graph is started, the above  
22 described counter insertion portion inserts into the insertion  
23 position the counter for determining the numbers of times of  
24 execution of the execution paths in both the above described  
25 in-outer loop structure graph and the above described in-inner  
26 loop structure graph.

27 (Article 12) The compiler apparatus according to article 10,  
28 wherein in the case where the insertion position in the above  
29 described program for inserting the counter for determining the  
30 number of times of execution of each execution path in the  
31 above described in-outer loop structure graph is the same as  
32 the position in the above described program for inserting the

1 counter for determining the number of times of execution of  
2 each execution path in the above described in-inner loop  
3 structure graph and then the counter of one, at the most, of  
4 the above described in-outer loop structure graph and the above  
5 described in-inner loop structure graph is started, the above  
6 described counter insertion portion generates a plurality of  
7 determination processes for determining the number of times of  
8 execution of each execution path in each of the above described  
9 in-outer loop structure graph and the above described in-inner  
10 loop structure graph, and the above described in-loop process  
11 frequency collection portion inserts the jump instruction for  
12 moving the process to another portion into the above described  
13 insertion position and sets the jump destination of the jump  
14 instruction at one of the above described plurality of  
15 determination processes so as to determine the number of times  
16 of execution of the execution paths in both the above described  
17 in-outer loop structure graph and the above described in-inner  
18 loop structure graph.

19 (Article 13) A compiler program for causing a computer to  
20 function as a compiler apparatus for collecting the frequencies  
21 with which each process is executed in the program to be  
22 optimized and optimizing the above described program based on  
23 the collected frequencies, the above described program causing  
24 the above described computer to function as the loop process  
25 detection portion for detecting the repeatedly executed loop  
26 process of the above described program, a loop process  
27 frequency collection portion for collecting the loop process  
28 frequencies with which the above described loop process is  
29 executed in the above described program, an in-loop process  
30 frequency collection portion for collecting in-loop process  
31 frequencies with which, as against the number of times of  
32 execution of the above described loop process, each of the  
33 plurality of in-loop processes included in the above described

1 loop process is executed; the in-loop execution information  
2 generating portion for, based on the above described loop  
3 process frequencies and the above described in-loop process  
4 frequencies, generating the in-loop execution information  
5 indicating the frequencies with which each of the above  
6 described plurality of in-loop processes is executed in the  
7 case where the above described program is executed, and the  
8 optimization portion for optimizing the above described program  
9 based on the above described in-loop execution information  
10 generated by the above described in-loop execution information  
11 generating portion.

12 (Article 14) The record medium having the compiler program  
13 according to article 13 recorded thereon.

14 (Article 15) A compilation method for collecting frequencies  
15 with which each process is executed in the program to be  
16 optimized and optimizing the above described program based on  
17 the collected frequencies, the above described method having  
18 the loop process detection step of detecting the repeatedly  
19 executed loop process of the above described program, a loop  
20 process frequency collection step of collecting the loop  
21 process frequencies with which the above described loop process  
22 is executed in the above described program, an in-loop process  
23 frequency collection step of collecting the in-loop process  
24 frequencies with which, as against the number of times of  
25 execution of the above described loop process, each of a  
26 plurality of in-loop processes included in the above described  
27 loop process is executed, an in-loop execution information  
28 generating step of, based on the above described loop process  
29 frequencies and the above described in-loop process  
30 frequencies, generating the in-loop execution information  
31 indicating the frequencies with which each of the above  
32 described plurality of in-loop processes is executed in the

1 case where the above described program is executed, and an  
2 optimization step of optimizing the above described program  
3 based on the above described in-loop execution information  
4 generated by the above described in-loop execution information  
5 generating portion.

6 (Article 16) A runtime information generating apparatus for  
7 collecting the frequencies with which each process is executed  
8 in the program to be optimized, the above described apparatus  
9 having the loop process detection portion for detecting the  
10 repeatedly executed loop process of the above described  
11 program, a loop process frequency collection portion for  
12 collecting the loop process frequencies with which the above  
13 described loop process is executed in the above described  
14 program, an in-loop process frequency collection portion for  
15 collecting the in-loop process frequencies with which, as  
16 against the number of times of execution of the above described  
17 loop process, each of a plurality of in-loop processes included  
18 in the above described loop process is executed, the in-loop  
19 execution information generating portion for, based on the  
20 above described loop process frequencies and the above  
21 described in-loop process frequencies, generating the in-loop  
22 execution information indicating the frequencies with which  
23 each of the above described plurality of in-loop processes is  
24 executed in the case where the above described program is  
25 executed, and optimizing the above described program based on  
26 the above described in-loop execution information generated by  
27 the above described in-loop execution information generating  
28 portion.

29 (Article 17) A runtime information generating program for  
30 causing a computer to function as the runtime information  
31 generating apparatus for collecting the frequencies with which  
32 each process is executed in the program to be optimized, the

1 above described program causing the above described computer to  
2 function as a loop process detection portion for detecting a  
3 repeatedly executed loop process of the above described  
4 program, a loop process frequency collection portion for  
5 collecting the loop process frequencies with which the above  
6 described loop process is executed in the above described  
7 program, an in-loop process frequency collection portion for  
8 collecting the in-loop process frequencies with which, as  
9 against the number of times of execution of the above described  
10 loop process, each of the plurality of in-loop processes  
11 included in the above described loop process is executed, and  
12 an in-loop execution information generating portion for, based  
13 on the above described loop process frequencies and the above  
14 described in-loop process frequencies, generating the in-loop  
15 execution information indicating the frequencies with which  
16 each of the above described plurality of in-loop processes is  
17 executed in the case where the above described program is  
18 executed, and causing the above described program to be  
19 optimized based on the above described in-loop execution  
20 information generated by the above described in-loop execution  
21 information generating portion.

22 (Article 18) The record medium having the runtime information  
23 generating program according to article 17 recorded thereon.

24 Advantages of the invention

25 As is clear from the above description, it is possible,  
26 according to the present invention, to collect the frequencies  
27 with which the processes of the program are executed at high  
28 speed.

29 Variations described for the present invention can be realized  
30 in any combination desirable for each particular application.



1 Thus particular limitations, and/or embodiment enhancements  
2 described herein, which may have particular advantages to a  
3 particular application need not be used for all applications.  
4 Also, not all limitations need be implemented in methods,  
5 systems and/or apparatus including one or more concepts of the  
6 present invention.

7 The present invention can be realized in hardware, software, or  
8 a combination of hardware and software. A visualization tool  
9 according to the present invention can be realized in a  
10 centralized fashion in one computer system, or in a distributed  
11 fashion where different elements are spread across several  
12 interconnected computer systems. Any kind of computer system -  
13 or other apparatus adapted for carrying out the methods and/or  
14 functions described herein - is suitable. A typical  
15 combination of hardware and software could be a general purpose  
16 computer system with a computer program that, when being loaded  
17 and executed, controls the computer system such that it carries  
18 out the methods described herein. The present invention can  
19 also be embedded in a computer program product, which comprises  
20 all the features enabling the implementation of the methods  
21 described herein, and which - when loaded in a computer system  
22 - is able to carry out these methods.

23 Computer program means or computer program in the present  
24 context include any expression, in any language, code or  
25 notation, of a set of instructions intended to cause a system  
26 having an information processing capability to perform a  
27 particular function either directly or after conversion to  
28 another language, code or notation, and/or reproduction in a  
29 different material form.

30 Thus the invention includes an article of manufacture which  
31 comprises a computer usable medium having computer readable

1 program code means embodied therein for causing a function  
2 described above. The computer readable program code means in  
3 the article of manufacture comprises computer readable program  
4 code means for causing a computer to effect the steps of a  
5 method of this invention. Similarly, the present invention may  
6 be implemented as a computer program product comprising a  
7 computer usable medium having computer readable program code  
8 means embodied therein for causing a a function described  
9 above. The computer readable program code means in the  
10 computer program product comprising computer readable program  
11 code means for causing a computer to effect one or more  
12 functions of this invention. Furthermore, the present  
13 invention may be implemented as a program storage device  
14 readable by machine, tangibly embodying a program of  
15 instructions executable by the machine to perform method steps  
16 for causing one or more functions of this invention.

17 It is noted that the foregoing has outlined some of the more  
18 pertinent objects and embodiments of the present invention.  
19 This invention may be used for many applications. Thus,  
20 although the description is made for particular arrangements  
21 and methods, the intent and concept of the invention is  
22 suitable and applicable to other arrangements and applications.  
23 It will be clear to those skilled in the art that modifications  
24 to the disclosed embodiments can be effected without departing  
25 from the spirit and scope of the invention. The described  
26 embodiments ought to be construed to be merely illustrative of  
27 some of the more prominent features and applications of the  
28 invention. Other beneficial results can be realized by  
29 applying the disclosed invention in a different manner or  
30 modifying the invention in ways known to those familiar with  
31 the art.

32 Variations described for the present invention can be realized

1 in any combination desirable for each particular application.  
2 Thus particular limitations, and/or embodiment enhancements  
3 described herein, which may have particular advantages to the  
4 particular application need not be used for all applications.  
5 Also, not all limitations need be implemented in methods,  
6 systems and/or apparatus including one or more concepts of the  
7 present invention.

8 The present invention can be realized in hardware, software, or  
9 a combination of hardware and software. A visualization tool  
10 according to the present invention can be realized in a  
11 centralized fashion in one computer system, or in a distributed  
12 fashion where different elements are spread across several  
13 interconnected computer systems. Any kind of computer system -  
14 or other apparatus adapted for carrying out the methods and/or  
15 functions described herein - is suitable. A typical  
16 combination of hardware and software could be a general purpose  
17 computer system with a computer program that, when being loaded  
18 and executed, controls the computer system such that it carries  
19 out the methods described herein. The present invention can  
20 also be embedded in a computer program product, which comprises  
21 all the features enabling the implementation of the methods  
22 described herein, and which - when loaded in a computer system  
23 - is able to carry out these methods.

24 Computer program means or computer program in the present  
25 context include any expression, in any language, code or  
26 notation, of a set of instructions intended to cause a system  
27 having an information processing capability to perform a  
28 particular function either directly or after conversion to  
29 another language, code or notation, and/or reproduction in a  
30 different material form.

31 Thus the invention includes an article of manufacture which

1 comprises a computer usable medium having computer readable  
2 program code means embodied therein for causing a function  
3 described above. The computer readable program code means in  
4 the article of manufacture comprises computer readable program  
5 code means for causing a computer to effect the steps of a  
6 method of this invention. Similarly, the present invention may  
7 be implemented as a computer program product comprising a  
8 computer usable medium having computer readable program code  
9 means embodied therein for causing a a function described  
10 above. The computer readable program code means in the  
11 computer program product comprising computer readable program  
12 code means for causing a computer to effect one or more  
13 functions of this invention. Furthermore, the present  
14 invention may be implemented as a program storage device  
15 readable by machine, tangibly embodying a program of  
16 instructions executable by the machine to perform method steps  
17 for causing one or more functions of this invention.

18 It is noted that the foregoing has outlined some of the more  
19 pertinent objects and embodiments of the present invention.  
20 This invention may be used for many applications. Thus,  
21 although the description is made for particular arrangements  
22 and methods, the intent and concept of the invention is  
23 suitable and applicable to other arrangements and applications.  
24 It will be clear to those skilled in the art that modifications  
25 to the disclosed embodiments can be effected without departing  
26 from the spirit and scope of the invention. The described  
27 embodiments ought to be construed to be merely illustrative of  
28 some of the more prominent features and applications of the  
29 invention. Other beneficial results can be realized by  
30 applying the disclosed invention in a different manner or  
31 modifying the invention in ways known to those familiar with  
32 the art.